


CS 525: Advanced Database Organization

06: Even more index structures

Boris Glavic



Slides: adapted from a [course](#) taught by [Hector Garcia-Molina](#), Stanford InfoLab

CS 525 COMPUTER SCIENCE Notes 6 - More Indices 1 IIT College of Science and Letters ILLINOIS INSTITUTE OF TECHNOLOGY

Recap

- We have discussed
 - Conventional Indices
 - B-trees
 - Hashing
 - Trade-offs
 - Multi-key indices
 - Multi-dimensional indices
 - ... but no example

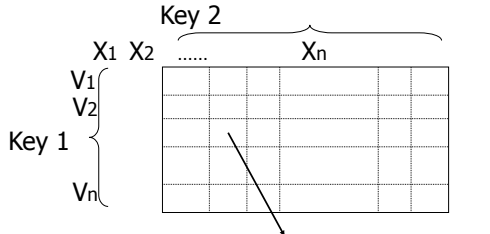
CS 525 COMPUTER SCIENCE Notes 6 - More Indices 2 IIT College of Science and Letters ILLINOIS INSTITUTE OF TECHNOLOGY

Today

- Multi-dimensional index structures
 - kd-Trees (very similar to example before)
 - **Grid File (Grid Index)**
 - Quad Trees
 - **R Trees**
 - **Partitioned Hash**
 - ...
- **Bitmap-indices**
- **Tries**

CS 525 COMPUTER SCIENCE Notes 6 - More Indices 3 IIT College of Science and Letters ILLINOIS INSTITUTE OF TECHNOLOGY

Grid Index



To records with key1=V3, key2=X2

CS 525 COMPUTER SCIENCE Notes 5 - Hashing 4 IIT College of Science and Letters ILLINOIS INSTITUTE OF TECHNOLOGY

CLAIM

- Can quickly find records with
 - key 1 = $V_i \wedge$ Key 2 = X_j
 - key 1 = V_i
 - key 2 = X_j

CS 525 COMPUTER SCIENCE Notes 5 - Hashing 5 IIT College of Science and Letters ILLINOIS INSTITUTE OF TECHNOLOGY

CLAIM

- Can quickly find records with
 - key 1 = $V_i \wedge$ Key 2 = X_j
 - key 1 = V_i
 - key 2 = X_j
- And also ranges....
 - E.g., key 1 $\geq V_i \wedge$ key 2 $< X_j$

CS 525 COMPUTER SCIENCE Notes 5 - Hashing 6 IIT College of Science and Letters ILLINOIS INSTITUTE OF TECHNOLOGY

• How do we find entry i, j in linear structure?

max number of i values $N=4$

$pos(i, j) =$

0,0	position S+0
0,1	position S+1
0,2	position S+2
0,3	position S+3
1,0	position S+4
1,1	
1,2	
1,3	
2,0	
2,1	position S+9
2,2	
2,3	
3,0	

CS 525 COMPUTER SCIENCE Notes 5 - Hashing 7 IIT College of Science and Letters ILLINOIS INSTITUTE OF TECHNOLOGY

• How do we find entry i, j in linear structure?

max number of i values $N=4$

$pos(i, j) = S + iN + j$

Issue: Cells must be same size, and N must be constant!

Issue: Some cells may overflow, some may be sparse...

0,0	position S+0
0,1	position S+1
0,2	position S+2
0,3	position S+3
1,0	position S+4
1,1	
1,2	
1,3	
2,0	
2,1	position S+9
2,2	
2,3	
3,0	

CS 525 COMPUTER SCIENCE Notes 5 - Hashing 8 IIT College of Science and Letters ILLINOIS INSTITUTE OF TECHNOLOGY

Solution: Use Indirection

Buckets

V1 X1 X2 X3

V2

V3

V4

* only pointers to buckets

Buckets

CS 525 COMPUTER SCIENCE Notes 5 - Hashing 9 IIT College of Science and Letters ILLINOIS INSTITUTE OF TECHNOLOGY

With indirection:

- Grid can be regular without wasting space
- We do have price of indirection

CS 525 COMPUTER SCIENCE Notes 5 - Hashing 10 IIT College of Science and Letters ILLINOIS INSTITUTE OF TECHNOLOGY

Can also index grid on value ranges

Salary Grid

0-20K	1
20K-50K	2
50K-∞	3

Linear Scale

1	2	3
Toy	Sales	Personnel

CS 525 COMPUTER SCIENCE Notes 5 - Hashing 11 IIT College of Science and Letters ILLINOIS INSTITUTE OF TECHNOLOGY

Grid files

- ⊕ Good for multiple-key search
- ⊖ Space, management overhead (nothing is free)
- ⊖ Need partitioning ranges that evenly split keys

CS 525 COMPUTER SCIENCE Notes 5 - Hashing 12 IIT College of Science and Letters ILLINOIS INSTITUTE OF TECHNOLOGY

Partitioned hash function

Idea:

010110 1110010

Key1 → h1 ← Key2 → h2 ←

CS 525 COMPUTER SCIENCE Notes 5 - Hashing 13 IIT College of Science and Letters ILLINOIS INSTITUTE OF TECHNOLOGY

EX:

h1(toy) = 0	000	
h1(sales) = 1	001	
h1(art) = 1	010	
.	011	
.	100	
h2(10k) = 01	101	
h2(20k) = 11	110	
h2(30k) = 01	111	
h2(40k) = 00		

Insert → <Fred,toy,10k>, <Joe,sales,10k>
<Sally,art,30k>

CS 525 COMPUTER SCIENCE Notes 5 - Hashing 14 IIT College of Science and Letters ILLINOIS INSTITUTE OF TECHNOLOGY

EX:

h1(toy) = 0	000	
h1(sales) = 1	001	<Fred>
h1(art) = 1	010	
.	011	
.	100	
h2(10k) = 01	101	
h2(20k) = 11	110	<Joe><Sally>
h2(30k) = 01	111	
h2(40k) = 00		

Insert → <Fred,toy,10k>, <Joe,sales,10k>
<Sally,art,30k>

CS 525 COMPUTER SCIENCE Notes 5 - Hashing 15 IIT College of Science and Letters ILLINOIS INSTITUTE OF TECHNOLOGY

EX:

h1(toy) = 0	000	<Fred>
h1(sales) = 1	001	<Joe><Jan>
h1(art) = 1	010	<Mary>
.	011	
.	100	<Sally>
h2(10k) = 01	101	
h2(20k) = 11	110	<Tom><Bill>
h2(30k) = 01	111	<Andy>
h2(40k) = 00		

Find Emp. with Dept. = Sales \wedge Sal=40k

CS 525 COMPUTER SCIENCE Notes 5 - Hashing 16 IIT College of Science and Letters ILLINOIS INSTITUTE OF TECHNOLOGY

EX:

h1(toy) = 0	000	<Fred>
h1(sales) = 1	001	<Joe><Jan>
h1(art) = 1	010	<Mary>
.	011	
.	100	<Sally>
h2(10k) = 01	101	
h2(20k) = 11	110	<Tom><Bill>
h2(30k) = 01	111	<Andy>
h2(40k) = 00		

Find Emp. with Dept. = Sales \wedge Sal=40k

CS 525 COMPUTER SCIENCE Notes 5 - Hashing 17 IIT College of Science and Letters ILLINOIS INSTITUTE OF TECHNOLOGY

EX:

h1(toy) = 0	000	<Fred>
h1(sales) = 1	001	<Joe><Jan>
h1(art) = 1	010	<Mary>
.	011	
.	100	<Sally>
h2(10k) = 01	101	
h2(20k) = 11	110	<Tom><Bill>
h2(30k) = 01	111	<Andy>
h2(40k) = 00		



Find Emp. with Sal=30k

CS 525 COMPUTER SCIENCE Notes 5 - Hashing 18 IIT College of Science and Letters ILLINOIS INSTITUTE OF TECHNOLOGY

EX:

h1(toy) =0	000	<Fred>
h1(sales) =1	001	<Joe><Jan>
h1(art) =1	010	<Mary>
.	011	
.	100	<Sally>
h2(10k) =01	101	
h2(20k) =11	110	<Tom><Bill>
h2(30k) =01	111	<Andy>
h2(40k) =00		



Find Emp. with Sal=30k

CS 525  Notes 5 - Hashing 19 

EX:

h1(toy) =0	000	<Fred>
h1(sales) =1	001	<Joe><Jan>
h1(art) =1	010	<Mary>
.	011	
.	100	<Sally>
h2(10k) =01	101	
h2(20k) =11	110	<Tom><Bill>
h2(30k) =01	111	<Andy>
h2(40k) =00		



Find Emp. with Dept. = Sales

CS 525  Notes 5 - Hashing 20 

EX:



h1(toy) =0	000	<Fred>
h1(sales) =1	001	<Joe><Jan>
h1(art) =1	010	<Mary>
.	011	
.	100	<Sally>
h2(10k) =01	101	
h2(20k) =11	110	<Tom><Bill>
h2(30k) =01	111	<Andy>
h2(40k) =00		



Find Emp. with Dept. = Sales

CS 525  Notes 5 - Hashing 21 

R-tree



- Nodes can store up to **M** entries
 - Minimum fill requirement (depends on variant)
- Each node rectangle in **n**-dimensional space
 - Minimum Bounding Rectangle (MBR) of its children
- MBRs of siblings are allowed to overlap
 - Different from B-trees
- balanced

CS 525  Notes 6 - More Indices 22 

CS 525  Notes 6 - More Indices 23 

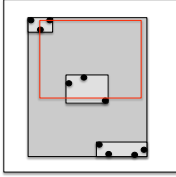
R-tree - Search



- Point Search
 - Search for $p = \langle x_i, y_i \rangle$
 - Keep list of potential nodes
 - Needed because of overlap
 - Traverse to child if MBR of child contains p

CS 525  Notes 6 - More Indices 24 

R-tree - Search

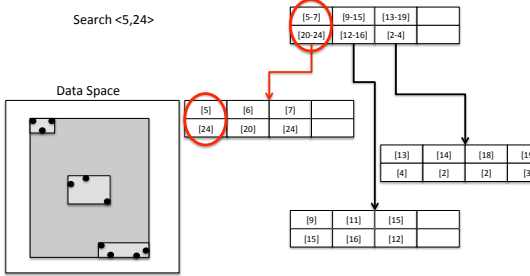
- Point Search
 - Search for points in region = $\langle [x_{min} - x_{max}], [y_{min} - y_{max}] \rangle$
 - Keep list of potential nodes
 - Traverse to child if MBR of child overlaps with query region





CS 525

Notes 6 - More Indices 25


R-tree - Search



Search $\langle 5, 24 \rangle$



CS 525

Notes 6 - More Indices 26




R-tree - Insert

- Similar to B-tree, but more complex
 - Overlap -> multiple choice where to add entry
 - Split harder because more choice how to split node (compare B-tree = 1 choice)
- 1) Find potential subtrees for current node
 - Choose one for insert (heuristic, e.g., the one the would grow the least)
 - Continue until leaf is found

CS 525

Notes 6 - More Indices 27




R-tree - Insert

- 2) Insert into leaf
- 3) Leaf is full? -> split
 - Find best split (minimum overlap between new nodes) is hard ($O(2^M)$)
 - Use linear or quadratic heuristics (original paper)
- 4) Adapt parents if necessary

CS 525

Notes 6 - More Indices 28




R-tree - Delete

- 1) Find leaf node that contains entry
- 2) Delete entry
- 3) Leaf node underflow?
 - Remove leaf node and cache entries
 - Adapt parents
 - Reinsert deleted entries

CS 525

Notes 6 - More Indices 29




Bitmap Index

- Domain of values $D = \{d_1, \dots, d_n\}$
 - Gender {male, female}
 - Age {1, ..., 120?}
- Use one vector of bits for each value
 - One bit for each record
 - 0: record has different value in this attribute
 - 1: record has this value

CS 525

Notes 6 - More Indices 30


Bitmap Index Example

Age			Todlers			Gender	
1	2	3	Name	Age	Gender	male	female
1	0	0	Peter	1	male	1	0
0	1	0	Gertrud	2	female	0	1
1	0	0	Joe	1	male	1	0
0	0	1	Marry	3	female	0	1



CS 525  Notes 6 - More Indices 31 

Bitmap Index Example

Age			Todlers			Gender	
1	2	3	Name	Age	Gender	male	female
1	0	0	Peter	1	male	1	0
0	1	0	Gertrud	2	female	0	1
1	0	0	Joe	1	male	1	0
0	0	1	Marry	3	female	0	1

Find all todlers with age 2 and sex female:
Bitwise-and between vectors

0
1
0
0



CS 525  Notes 6 - More Indices 32 

Bitmap Index Example

Age			Todlers			Gender	
1	2	3	Name	Age	Gender	male	female
1	0	0	Peter	1	male	1	0
0	1	0	Gertrud	2	female	0	1
1	0	0	Joe	1	male	1	0
0	0	1	Marry	3	female	0	1



Find all todlers with age 2 or sex female:
Bitwise-or between vectors

0
1
0
1

CS 525  Notes 6 - More Indices 33 



Compression

- Observation:
 - Each record has one value in index attribute
 - For N records and domain of size |D|
 - Only 1/|D| bits are 1
 - > waste of space
- Solution
 - Compress data
 - Need to make sure that **and** and **or** is still fast

CS 525  Notes 6 - More Indices 34 



Run length encoding (RLE)

- Instead of actual 0-1 sequence encode length of 0 or 1 runs
- One bit to indicate whether 0/1 run + several bits to encode run length
- But how many bits to use to encode a run length?
 - Gamma codes or similar to have variable number of bits

CS 525  Notes 6 - More Indices 35 

RLE Example

- 0001 0000 1110 1111 (2 bytes)
- 3, 1,4, 3, 1,4 (6 bytes)
- -> if we use one byte to encode a run we have 7 bits for length = max run length is 128(127)

CS 525  Notes 6 - More Indices 36 

Elias Gamma Codes

- $X = 2^N + (x \bmod 2^N)$
 - Write N as N zeros followed by one 1
 - Write $(x \bmod 2^N)$ as N bit number
- $18 = 2^4 + 2 = 000010010$
- 0001 0000 1110 1111 **(2 bytes)**
- 3, 1,4, 3, 1,4 **(6 bytes)**
- 0111 0010 0011 1001 00 **(3 bytes)**

CS 525
Notes 6 - More Indices 37

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Hybrid Encoding

- Run length encoding
 - Can waste space
 - And/or run length not aligned to byte/word boundaries
- Encode some bytes of sequence as is and only store long runs as run length
 - EWAH
 - BBC (that's what Oracle uses)

CS 525
Notes 6 - More Indices 38

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Extended Word aligned Hybrid (EWAH)

- Segment sequence in machine words (64bit)
- Use two types of words to encode
 - Literal words, taken directly from input sequence
 - Run words
 - ½ word is used to encode a run
 - ½ word is used to encode how many literals follow

0000 0000	0000 0000	0010 1000	1111 1111	1100 0010
0010 0001	0010 1000	1001 0001	1100 0010	

CS 525
Notes 6 - More Indices 39

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Bitmap Indices

- Fast for read intensive workloads
 - Used a lot in datawarehousing
- Often build on the fly during query processing
 - As we will see later in class

CS 525
Notes 6 - More Indices 40

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Trie

- From Retrieval
- Tree index structure
- Keys are sequences of values from a domain D
 - $D = \{0,1\}$
 - $D = \{a,b,c,\dots,z\}$
- Key size may or may not be fixed
 - Store 4-byte integers using $D = \{0,1\}$ (32 elements)
 - Strings using $D = \{a,\dots,z\}$ (arbitrary length)

CS 525
Notes 6 - More Indices 41

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Trie

- Each node has pointers to $|D|$ child nodes
 - One for each value of D
- Searching for a key $k = [d_1, \dots, d_n]$
 - Start at the root
 - Follow child for value d_i

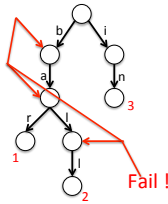
CS 525
Notes 6 - More Indices 42

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Trie Example

Words: bar, ball, in

Search for **bald**



CS 525



Notes 6 - More Indices

43

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Tries Implementation

- 1) Each node has an array of child pointers
- 2) Each node has a list of hash table of child pointers
- 3) array compression schemes derived from compressed DFA representations

CS 525



Notes 6 - More Indices

44

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Summary

Discussion:

- Conventional Indices
- B-trees
- Hashing (extensible, linear)
- SQL Index Definition
- Index vs. Hash
- Multiple Key Access
- Multi Dimensional Indices
 - Variations: Grid, R-tree,
- Partitioned Hash
- Bitmap indices and compression
- Tries

CS 525



Notes 5 - Hashing

45

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY